This is the post peer-review accepted manuscript of:

I. Notarnicola and G. Notarstefano, "A randomized primal distributed algorithm for partitioned and big-data non-convex optimization," 2016 IEEE 55th Conference on Decision and Control (CDC), Las Vegas, NV, 2016, pp. 153-158.

The published version is available online at:

https://doi.org/10.1109/CDC.2016.7798262

# A randomized primal distributed algorithm for partitioned and big-data non-convex optimization

Ivano Notarnicola and Giuseppe Notarstefano

***Abstract*— In this paper we consider a distributed optimization scenario in which the aggregate objective function to minimize is partitioned, big-data and possibly non-convex. Specifically, we focus on a set-up in which the dimension of the decision variable depends on the network size as well as the number of local functions, but each local function handled by a node depends only on a (small) portion of the entire optimization variable. This problem set-up has been shown to appear in many interesting network application scenarios. As main paper contribution, we develop a simple, primal distributed algorithm to solve the optimization problem, based on a randomized descent approach, which works under asynchronous gossip communication. We prove that the proposed asynchronous algorithm is a proper, ad-hoc version of a coordinate descent method and thus converges to a stationary point. To show the effectiveness of the proposed algorithm, we also present numerical simulations on a non-convex quadratic program, which confirm the theoretical results.**

***Index Terms*— primal, non-convex, proximal, asynchronous, randomized, coordinate, big-data, partitioned.**

## I. Introduction

In several network scenarios optimization problems arise in which an aggregate cost function, sum of local cost functions, needs to be minimized in a distributed way. A typical approach in distributed optimization is to develop algorithms in which the processors in the network reach consensus on a minimizer of the problem. However, when the dimension of the decision variable depends on the number of agents in the network the consensus approach gives rise to algorithms which scale badly with the network size. Enforcing consensus on the entire vector of decision variables is not necessary in many important applications, since the nodes are interested in computing only part of the decision vector, namely only some local variables of interest. In this paper we consider a partitioned problem set-up in which the aggregate function is the sum of local functions, each one depending only on a portion of the decision vector. For this set-up our goal is to design a distributed algorithm in which the nodes compute only a local portion of interest of the entire solution vector, so that the whole minimizer can be obtained by stacking together the local portions.

This partitioned set-up has been introduced in [1] where a distributed ADMM-based algorithm is proposed. In [2] some concrete motivating scenarios are described for the same set-up and a dual decomposition algorithm is proposed. In both the above references the algorithms are designed for a synchronous network with a fixed communication graph. In [3], an analogous problem formulation is considered within a parallel context. The authors propose a coordinate descent method and derive its convergence rate. In [4] the authors propose a distributed algorithm for a partitioned quadratic program under lossy communication. A distributed ADMM-based algorithm with applications in MPC is proposed in [5] to deal with an unconstrained optimization problem with local domains which is related to the set-up in this paper.

Usually, distributed approaches need a common clock (e.g., because a diminishing (time-varying) step-size is used). We want to avoid this limitation designing an asynchronous, event-triggered protocol based on local and independent timers, [6]. A Newton-Raphson consensus strategy is proposed in [7] to solve unconstrained, convex optimization problems under asynchronous, symmetric gossip communications. In [8] a self-triggered communication protocol is considered. Based on an error condition a distributed, continuous-time algorithm is developed. In [9] an asynchronous ADMM-based distributed method is proposed for a separable, constrained optimization problem with a convergence rate $O(1/t)$. A distributed, asynchronous algorithm for constrained optimization based on random projections is proposed in [10].

The asynchronous, distributed algorithm we design in this paper is based on a (randomized) coordinate descent method. In [11] the coordinate method for huge scale optimization has been introduced. This powerful approach has been extended to deal with (convex) composite objective functions and parallel scenarios, see [3], [12], [13]. In [14] a coordinate approach to solve linearly constrained problems has been proposed. Using a coordinate ADMM-based approach, in [15] a distributed, asynchronous algorithm is developed.

Regarding non-convex optimization problems, in [16], the authors extend the coordinate approach to large-scale non-convex optimization proving the rate of convergence of their algorithms. A parallel algorithm based on local strongly convex approximations is exploited in [17] to cope with non-convex optimization problems. The latter approach has been extended to a distributed context in [18]. In [19], the authors proposed an auction-based distributed algorithm for non-convex optimization.

As main paper contribution we propose an asynchronous, distributed algorithm to solve partitioned, big-data non-convex optimization problems. The proposed primal algorithm is based on local updates involving the minimization of a strongly convex, quadratic approximation of the objec-

Ivano Notarnicola and Giuseppe Notarstefano are with the Department of Engineering, Università del Salento, Via Monteroni, 73100 Lecce, Italy, name.lastname@unisalento.it. This result is part of a project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 638992 - OPT4SMART).

tive function. Each node constructs this approximation by exchanging information only with neighboring nodes. The updates at each node are regulated by a local timer that triggers independently from the ones of the other nodes. We prove the convergence in probability of the distributed algorithm by showing that it is equivalent to a generalized coordinate descent method for the minimization of non-convex composite functions. The generalized coordinate descent algorithm extends the one proposed in [16] and thus represents a side interesting result.

The paper is organized as follows. In Section II we present the problem set-up. In Section III we propose our algorithm and prove its convergence in Section IV. Finally, in Section V we show some simulations.

*Notation:* Consider a vector $x \in \mathbb{R}^n$ partitioned in $N$ block-components as follows

$$x = [x_1^\top, \dots, x_N^\top]^\top, \tag{1}$$

where, for all $i \in \{1, \dots, N\}$, we have $x_i \in \mathbb{R}^{n_i}$ and $\sum_{i=1}^N n_i = n$. Moreover, consider a block decomposition of the $N \times N$ identity matrix $I = [U_1, \dots, U_N]$, where for all $i \in \{1, \dots, N\}$ each $U_i \in \mathbb{R}^{n \times n_i}$. Then we can write $x_i = U_i^\top x$ and $x = \sum_{i=1}^N U_i x_i$. For a function $\varphi : \mathbb{R}^n \to \mathbb{R}$, we denote $\nabla_{x_i} \varphi(\bar{x}) = U_i^\top \nabla \varphi(\bar{x})$ the "partial" gradient of $\varphi$ with respect to $x_i \in \mathbb{R}^{n_i}$.

## II. Optimization problem set-up

We consider a network of $N$ nodes which can interact according to a fixed, undirected communication graph $\mathcal{G} = (\{1, \dots, N\}, \mathcal{E})$, where $\mathcal{E} \subseteq \{1, \dots, N\} \times \{1, \dots, N\}$ is the set of edges. That is, the edge $(i, j)$ models the fact that node $i$ and $j$ can exchange information. We denote by $\mathcal{N}_i$ the set of *neighbors* of node $i$ in the fixed graph $\mathcal{G}$, i.e., $\mathcal{N}_i := \{j \in \{1, \dots, N\} \mid (i, j) \in \mathcal{E}\}$, and by $|\mathcal{N}_i|$ its cardinality. Here we assume that the graph contains also self-edges, so that $\mathcal{N}_i$ contains also $i$.

We want to stress that the fixed graph only models, for each node, the set of possible neighbors the node can communicate with. On top of this graph, we will consider an asynchronous communication protocol described later.

We start by a common set-up in distributed optimization, i.e., the minimization of a separable cost function composed by two contributions, i.e., $\min_{x \in \mathbb{R}^n} \sum_{i=1}^N f_i(x) + g_i(x)$, where $f_i : \mathbb{R}^n \to \mathbb{R}$ and $g_i : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$, with $N, n \in \mathbb{N}$. Usually this composite structure of the objective functions, is used to split the effective cost into a smooth part (modeling some local objective) and a (possibly) non-smooth one being a regularization term or a constraint.[1]

In this paper we consider problems in which the composite function has a *partitioned structure*, that we next describe. Let the decision variable $x \in \mathbb{R}^n$ be partitioned as stated in (1), then the sub-vector $x_i \in \mathbb{R}^{n_i}$ with $n_i \ll n$, represents the relevant information at node $i$. Each local objective $f_i$ has a sparsity consistent with the interaction graph, namely, for $i \in \{1, \dots, N\}$, the function $f_i$ depends only on the component of node $i$ and of its neighbors. To highlight this property we let $f_i : \mathbb{R}^{\sum_{j \in \mathcal{N}_i} n_j} \to \mathbb{R}$ and write $f_i(x_{\mathcal{N}_i})$. Also, each function $g_i$ depends only on the component $x_i$, i.e., $g_i : \mathbb{R}^{n_i} \to \mathbb{R} \cup \{+\infty\}$.

In light of the described structure, the problem we aim at solving in a distributed way can be written as

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^N f_i(x_{\mathcal{N}_i}) + g_i(x_i), \tag{2}$$

where node $i$ knows only the functions $f_i$ and $g_i$. We call this problem *partitioned* (due to the structure of the functions $f_i$ and $g_i$) and *big-data* (since the dimension of the decision variable depends on the number of nodes).

Note that, in this partitioned scenario, network structure and objective function are inherently related. That is, nodes that share a variable are neighbors in the communication graph. As pointed out in the introduction this set-up appears in several interesting applications [2]. In the following assumptions we state the main properties of problem (2).

*Assumption 2.1:* For all $i \in \{1, \dots, N\}$, $f_i$ is a smooth function of $x_{\mathcal{N}_i}$. In particular, $f_i$ has block-coordinate Lipschitz continuous gradient, i.e., for all $j \in \mathcal{N}_i$ there exists constants $L_{ij} > 0$ such that for all $x_{\mathcal{N}_i} \in \mathbb{R}^{\sum_{\ell \in \mathcal{N}_i} n_\ell}$ and $s_j \in \mathbb{R}^{n_j}$ it holds

$$\|\nabla_{x_j} f_i(x_{\mathcal{N}_i} + U_{ij} s_j) - \nabla_{x_j} f_i(x_{\mathcal{N}_i})\| \leq L_{ij} \|s_j\|.$$

where $U_{ij}$ is a suitable matrix such that $U_{ij} s_j$ is a vector in $\mathbb{R}^{\sum_{\ell \in \mathcal{N}_i} n_\ell}$ with $j$-th block-component equal to $s_j$ and all the other ones equal to zero. □

In light of Assumption 2.1, it is easy to show that the following lemma holds.

*Lemma 2.2:* Let Assumption 2.1 hold, then the aggregate function $f(x) := \sum_{i=1}^N f_i(x_{\mathcal{N}_i})$ has block-coordinate Lipschitz continuous gradient. In particular, for all $i \in \{1, \dots, N\}$, the partial gradient $\nabla_{x_i} f$ has Lipschitz constant given by $L_i := \sum_{j \in \mathcal{N}_i} L_{ij}$.

*Proof:* The proof follows straight by simply writing the norm of the aggregate cost $f$ and then bounding each term of its gradient by using its block Lipschitz constant. ■

*Remark 2.3:* Note that one can assume directly that $\nabla_{x_i} f$ is Lipschitz continuous, but while the condition we impose can be checked in a distributed way, the weaker one needs a global knowledge of the cost $f$. □

*Assumption 2.4:* For all $i \in \{1, \dots, N\}$, the function $g_i$ is a proper, closed, proper, convex function. □

We stress that we have not assumed any convexity condition on $f_i$, thus optimization problem (2) is non-convex in general. Finally, we state the following assumption which is quite standard for non-convex scenarios.

*Assumption 2.5:* The cost $V(x) := \sum_{i=1}^N f_i(x_{\mathcal{N}_i}) + g_i(x_i)$ of problem (2) is a coercive function. □

Assumption 2.5 guarantees that at least a local minimum for problem (2) exists.

Figure 1 visualizes the sparsity structure for a function partitioned according to a path graph of $N = 4$ nodes.

[1] A constraint $x \in \cap_{i \in \{1, \dots, N\}} X_i \subset \mathbb{R}^n$ is modeled by setting $g_i(x) = I_{X_i}(x)$, with $I_{X_i}(x) = 0$ $\forall x \in X_i$ and $I_{X_i}(x) = +\infty$ otherwise.

Each $i$-th column shows the variables on which $f_i$ depends, while along each $i$-th row it is possible to see in which functions a variable $x_i$ appears. It is worth noticing that the sparsity in the $i$-th row shows the consistency that needs to be maintained among neighboring nodes on variable $x_i$.

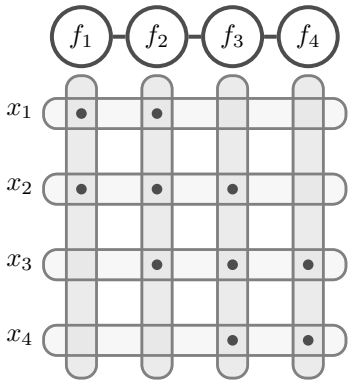


Fig. 1. Partitioned optimization problem over a path graph of $N=4$ nodes.

## III. Distributed optimization algorithm

In this section we present our asynchronous distributed algorithm.

In order to develop our algorithm, we need to introduce some technical tools: (i) the asynchronous communication protocol necessary to manage the overall behavior of the algorithm, and (ii) the local approximation model that each node will use to perform its local (descent) update.

We consider an asynchronous communication protocol where each node $i \in \{1,\dots,N\}$ has its own concept of time defined by a local timer $\tau_i$, which randomly and independently of the other nodes triggers when to awake itself. The timers trigger according to exponential distributions with a common parameter. We denote $T_i$ a realization drawn by node $i$. Between two triggering events the node is in an *idle* mode, i.e., it continuously receives messages from neighboring nodes and updates some internal variables. When a trigger occurs, it switches into an *awake* mode in which it updates its local variable and transmits the updated information to its neighbors. A formal discussion on this protocol is given in [6].

The proposed distributed algorithm is based on *local* quadratic, strongly-convex approximations of the cost function that each node computes.

Formally, each node $i \in \{1,\dots,N\}$ constructs the following local approximation of the entire cost function at a fixed $\bar{x} \in \mathbb{R}^n$ (neglecting the constant term $f(\bar{x})$ which does not affect the optimization),

$$q_i(s_i;\bar{x}) := \nabla_{x_i} f(\bar{x})^\top s_i + \frac{1}{2}\|s_i\|^2_{Q_i(\bar{x})} + g_i(\bar{x}_i + s_i) \qquad (3)$$
$$= \sum_{j\in\mathcal{N}_i} \nabla_{x_i} f_j(\bar{x}_{\mathcal{N}_j})^\top s_i + \frac{1}{2}\|s_i\|^2_{Q_i(\bar{x})} + g_i(\bar{x}_i + s_i)$$

with $Q_i(x) \in \mathbb{R}^{n_i \times n_i}$ a symmetric, positive definite matrix satisfying the following assumption.

*Assumption 3.1:* For any $x \in \mathbb{R}^n$ and $i \in \{1,\dots,N\}$ it holds that $Q_i(x) \succeq L_i I$. □

Intuitively Assumption 3.1 guarantees the strong convexity of $q_i$. The role of the Lipschitz constant $L_i$ in the bound will be clear in the analysis of the algorithm given in Section IV.

Informally, the asynchronous distributed optimization algorithms is as follows. A node $i$ takes care of modifying the variable $x_i$. We denote $\bar{x}_i$ the current state of node $i$, which is the estimated optimal value of the variable $x_i$. Consistently we denote $\bar{x}_{\mathcal{N}_i}$ the vector of states of nodes in $\mathcal{N}_i$.

When a node $i$ wakes up, it updates its state $\bar{x}_i$ by moving in the direction obtained from the minimization of its local approximation $q_i(s_i;\bar{x})$, being $\bar{x}$ the current value of the decision variable. Then, it sends to each neighbor $j \in \mathcal{N}_i$ the updated $x_i$ and $\nabla_{x_j} f_i(\bar{x}_{\mathcal{N}_i})$. When in idle, node $i$ is in a listening mode. If an updated $\nabla_{x_i} f_j(\bar{x}_{\mathcal{N}_j})$ is received from a neighbor $j$ no computation is needed. If $\bar{x}_j$ is also received ($j$ was an awake node) the following happens. Node $i$ updates the partial gradients of its local function $f_i$ according to the new $\bar{x}_j$, and sends back the updated partial gradients to its neighbors. In order to highlight the difference between updated and old variables at node $i$ during the awake phase, we denote the updated ones with a "+" symbol, e.g., as $\bar{x}_i^+$.

We want to stress two important aspects of the idle/awake cycle. First, these two phases are regulated by local timers without the need of any central clock. Second, when in idle a node only receives messages and from time to time evaluates a partial gradient, which takes a negligible time compared to the computation performed in the awake phase.

The distributed algorithm is formally reported in the table below (from the perspective of node $i$).

**Distributed Algorithm** Partitioned Coordinate Descent

**Processor state**: $\bar{x}_i$

**Initialization**: set $\tau_i = 0$ and get a realization $T_i$

**Evolution**:

***IDLE:***

WHILE: $\tau_i \le T_i$ DO:

receive $\bar{x}_j$ and/or $\nabla_{x_i} f_j(\bar{x}_{\mathcal{N}_j})$ from $j \in \mathcal{N}_i$

evaluate $\nabla_{x_j} f_i(\bar{x}_{\mathcal{N}_i})$ and send it to $j \in \mathcal{N}_i$

go to ***AWAKE***.

***AWAKE:***

compute $$d_i = \operatorname*{argmin}_{s_i} q_i(s_i;\bar{x}) \qquad (4)$$

update $$\bar{x}_i^+ = \bar{x}_i + d_i \qquad (5)$$

broadcast $\bar{x}_i^+$, $\nabla_{x_j} f_i(\bar{x}^+_{\mathcal{N}_i})$ to $j \in \mathcal{N}_i$

set $\tau_i = 0$, get a new realization $T_i$ and go to ***IDLE***.

We point out some aspects involving the local approximation (3) that each node uses in its local computations.

First, it is worth noting $q_i(s_i;\bar{x})$ does not depend on the entire state $\bar{x}$, but only on $\bar{x}_{\mathcal{N}_j}$, $j \in \mathcal{N}_i$ and therefore is

constructed by node $i$ by using only information from its neighbors. Moreover, node $i$ does not needed the expression of neighboring cost functions $f_j$ to build $q_i(s_i;\bar{x})$, but only the gradients $\nabla_{x_i} f_j$. In some special cases (discussed in the following paragraph), $Q_i(x)$ could include second order information of $f_j$, $j \in \mathcal{N}_i$, i.e., $\nabla^2_{x_i,x_i} f_j$, that should be sent together with the gradients.

Second, different choices for the weight matrix $Q_i(x)$ are allowed. By exploiting the block Lipschitz continuity of the gradient of $f$, a first simple choice is to set $Q_i(x) := L_i I$ for all $i \in \{1,\dots,N\}$ and $x \in \mathbb{R}^n$. Motivated by existing works in the literature, e.g., [17], non diagonal choices for $Q_i(x)$ are reasonable: for instance, assuming $f \in \mathcal{C}^2$, one can select a second order approximation, i.e., set $Q_i(x) := \nabla^2_{x_i,x_i} f(x) + \epsilon_i I$ for a sufficiently large $\epsilon_i > 0$ for all $i \in \{1,\dots,N\}$. As mentioned above this information can be constructed in a distributed manner.

Third and final, recalling the definition of the proximal operator $\mathbf{prox}_{\alpha,\varphi} : \mathbb{R}^n \to \mathbb{R}^n$ of a closed, proper, convex function $\varphi : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}$ given by $\mathbf{prox}_{\alpha,\varphi}(v) := \operatorname{argmin}_x \left(\varphi(x) + \frac{1}{2\alpha}\|x - v\|^2\right)$ with $\alpha > 0$, we have that for $Q_i(x) = L_i I$ the update law described in (4)-(5), can be rephrased in term of proximal operators and, thus, leading to a distributed coordinate proximal gradient method. On this regard it is worth noting that our algorithm, with a general expression for $Q_i$, can be written in terms of a generalized, weighted version of the proximal operator as follows. Given a positive definite matrix $W \in \mathbb{R}^{n\times n}$, we define

$$\mathbf{prox}_{W,\varphi}(v) := \operatorname*{argmin}_x \left\{\varphi(x) + \frac{1}{2}\big\|x - v\big\|^2_{W^{-1}}\right\}, \quad (6)$$

thus, the iteration (4)-(5) can be recast as

$$\bar{x}_i^+ = \mathbf{prox}_{Q_i(\bar{x})^{-1},g_i}\Big(\bar{x}_i - Q_i(\bar{x})^{-1} \sum_{j\in\mathcal{N}_i} \nabla_{x_i} f_j(\bar{x}_{\mathcal{N}_j})\Big).$$

## IV. Convergence analysis of the Partitioned Coordinate Descent distributed algorithm

In this section we prove the convergence in probability of the proposed algorithm.

First, it is worth pointing out that being the algorithm asynchronous, for the analysis we need to carefully formalize the concept of algorithm iterations. We will use a nonnegative integer variable $t$ indexing a change in the whole state $\bar{x} = [\bar{x}_1^\top \dots \bar{x}_N^\top]^\top$ of the distributed algorithm. In particular, each triggering will induce an *iteration* of the distributed optimization algorithm and will be indexed with $t$. We want to stress that this (integer) variable $t$ does not need to be known by the agents. That is, this timer is not a common clock and is only introduced for the sake of analysis.

*Theorem 4.1:* Let Assumptions 2.1, 2.4, 2.5 and 3.1 hold true. Then, the Partitioned Coordinate Descent distributed algorithm generates a sequence $x(t) := [\bar{x}_1(t)^\top,\dots,\bar{x}_N(t)^\top]^\top$ (obtained stacking the nodes' states) such that the random variable $V(x(t))$ converges almost surely, i.e., there exists a random variable $V^\star$ such that

$$\Pr\Big(V(x(t)) = V^\star\Big) = 1.$$

Moreover, any limit point $x^\star$ of $[\bar{x}_1(t)^\top,\dots,\bar{x}_N(t)^\top]^\top$ is a *stationary* point of problem (2) and, thus, satisfies its first order optimality condition, i.e., there exists a subgradient $\widetilde{\nabla} g(x^\star)$ of $g$ at $x^\star$ such that $\nabla f(x^\star) + \widetilde{\nabla} g(x^\star) = 0$. □

### A. Coordinate descent method for composite non-convex minimization

In this subsection we consider a more general composite optimization problem and prove a result that is instrumental to the convergence proof of our distributed algorithm. We introduce a generalization of the algorithm proposed in [13], [16], [20] based on the quadratic approximation introduced in (3). We present the algorithm for problem (2), but we want to stress that the algorithm can be applied to a general function $f : \mathbb{R}^n \to \mathbb{R}$ with block-Lipschitz continuous gradient. This will be clear from the analysis.

We consider a coordinate descent method based on selecting a random block-component, say $x_i$, of $x$ at each iteration and updating only $x_i$ through a suitable descent rule. The descent step is based on the quadratic approximation of the cost function given in (3). The coordinate descent method is formally summarized in the table below.

**Algorithm** Generalized Coordinate Descent Algorithm

Choose a random block $i_t \in \{1,\dots,N\}$ with probability $p_{i_t}$

Compute a descent direction $d_{i_t}$ solving

$$d_{i_t} = \operatorname*{argmin}_{s_i} q_{i_t}(s_i; x(t)) \quad (7)$$

Update the decision variable according to

$$\begin{aligned} x_{i_t}(t+1) &= x_{i_t}(t) + d_{i_t} \\ x_j(t+1) &= x_j(t), \qquad \text{for all } j \neq i_t \end{aligned} \quad (8)$$

In the following we present results for the theoretical convergence of the generalized coordinate descent algorithm.

*Lemma 4.2:* Let Assumption 2.1, 2.4, 3.1 hold. Let $x(t)$ be the random sequence generated by Generalized Coordinate Descent Algorithm, then for all $t \geq 0$ it holds

$$V(x(t+1)) \leq V(x(t)) - \frac{L_{i_t}}{2}\|d_{i_t}\|^2.$$

*Proof:* From Assumption 2.1 (Lipschitz continuity of $\nabla f$), we can write the well-known descent lemma (see [21, Proposition A.24]), for all $i \in \{1,\dots,N\}$ and for all $\bar{x} \in \mathbb{R}^n$

$$\begin{aligned} V(\bar{x} + U_i s_i) \leq\ & f(\bar{x}) + \nabla_{x_i} f(\bar{x})^\top s_i \\ & + \frac{L_i}{2}\|s_i\|^2 + g_i(\bar{x}_i + s_i) + \textstyle\sum_{j\neq i} g_j(\bar{x}_j), \end{aligned}$$

with $U_i$ introduced in the *Notation* paragraph.

Since $Q_i$ satisfies Assumption 3.1, then we can generalize the above descent condition by introducing a uniform bound depending on the Lipschitz constant of block $i$, i.e.,

$$\begin{aligned} V(\bar{x} + U_i s_i) \leq\ & f(\bar{x}) + \nabla_{x_i} f(\bar{x})^\top s_i \\ & + \frac{1}{2}\|s_i\|^2_{Q_i(\bar{x})} + g_i(\bar{x}_i + s_i) + \textstyle\sum_{j\neq i} g_j(\bar{x}_j) \end{aligned}$$

Due the partitioned structure of $f$, the explicit expression of $\nabla_{x_i} f(x)$ actually depends only on $f_j$, $j \in \mathcal{N}_i$, thus the latter condition can be further rephrased as

$$V(\bar{x} + U_i s_i) \leq q_i(s_i; \bar{x}) + f(\bar{x}) + \textstyle\sum_{j \neq i} g_j(\bar{x}_j). \quad (9)$$

with $q_i(s_i; \bar{x})$ defined as in (3).

Consider a descent direction $d_{i_t}$ computed as in (7), then $d_{i_t}$ satisfies the first order necessary condition of optimality for problem (7)

$$\nabla_{x_{i_t}} f(x(t)) + Q_{i_t}(x(t)) d_{i_t} + \widetilde{\nabla} g_{i_t}(x_{i_t}(t) + d_{i_t}) = 0, \quad (10)$$

where $\widetilde{\nabla} g_{i_t} \in \mathbb{R}^{m_{i_t}}$ is a particular subgradient of $g_{i_t}$.

Starting form equation (9) with the following identification $\bar{x} = x(t)$ and $\bar{x} + U_{i_t} d_{i_t} = x(t+1)$, and adding and subtracting the term $g_{i_t}(x_{i_t}(t))$ we obtain

$$\begin{aligned} V(x(t+1)) &\leq V(x(t)) + \nabla_{x_{i_t}} f(x(t))^\top d_{i_t} + \frac{1}{2} \|d_{i_t}\|^2_{Q_{i_t}(x(t))} \\ &\quad + g_{i_t}(x_{i_t}(t) + d_{i_t}) - g_{i_t}(x_{i_t}(t)) \\ &\leq V(x(t)) + \nabla_{x_{i_t}} f(x(t))^\top d_i \\ &\quad + \frac{1}{2} \|d_{i_t}\|^2_{Q_{i_t}(x(t))} + \widetilde{\nabla} g_{i_t}(x_{i_t}(t) + d_{i_t})^\top d_{i_t} \\ &\leq V(x(t)) - \frac{1}{2} \|d_{i_t}\|^2_{Q_{i_t}(x(t))} \\ &\leq V(x(t)) - \frac{L_i}{2} \|d_{i_t}\|^2 \end{aligned}$$

where we used the convexity of $g_{i_t}$, the optimality condition (10) and the uniform bound in Assumption 3.1. ∎

*Theorem 4.3:* Let Assumptions 2.1, 2.4, 2.5 and 3.1 hold true. Then, the Generalized Coordinate Descent Algorithm generates a sequence $x(t)$ such that the random variable $V(x(t))$ converges almost surely. Moreover, any limit point $x^\star$ of $x(t)$ is a stationary point of $V$ and, thus, satisfies the first order necessary condition for optimality for problem (2), i.e., there exists a subgradient $\widetilde{\nabla} g(x^\star)$ of $g$ at $x^\star$ such that

$$\nabla f(x^\star) + \widetilde{\nabla} g(x^\star) = 0$$

*Proof:* The result is proven by following the same line as in [16, Theorem 1] where the generalized Lemma 4.2 is used in place of [16, Lemma 3]. ∎

### B. Proof of Theorem 4.1

Our proof strategy is based on showing that the iterations of the asynchronous distributed algorithm can be written as the iterations of an ad-hoc version of the coordinate descent method for composite non-convex functions given in Section IV-A.

*Timer model and uniform node extraction.* Since the timers trigger independently according to the same exponential distribution, then from an external, global perspective, the induced awaking process of the nodes corresponds to the following: only one node per iteration wakes up randomly, uniformly and independently from previous iterations. Thus, each triggering, which induces an *iteration* of the distributed optimization algorithm and is indexed with $t$, corresponds to the (uniform) selection of a node in $\{1, \ldots, N\}$ that becomes awake. We denote $i_t$ the extracted node. Notice that node $i_t$ changes the value of its state $\bar{x}_{i_t}$ while all the other states are not changed by the algorithm.

*State consistency (inductive argument).* Next we show by induction that if all the nodes have a consistent and updated information before a node $i$ gets awake, then the same holds after the update. By consistent we mean that for a variable $x_\ell$, all the nodes in $\mathcal{N}_\ell$ have the same state $\bar{x}_\ell$. By updated we mean that each node $\ell$ has an updated value of the gradients $\nabla_{x_\ell} f_j$, $j \in \mathcal{N}_\ell$. First, node $i$ changes only its state $\bar{x}_i$ relative to the variable $x_i$. This variable is shared only with neighbors $j \in \mathcal{N}_i$, which receive the new state $\bar{x}_i$ after the update. As regards the gradients, the ones affected by the change of the variable $x_i$ are $\nabla_{x_i} f_j$, with $j \in \mathcal{N}_i$. Notice that these gradients are only used by nodes $k \in \mathcal{N}_j$. But after the broadcast performed by $i$, each idle $j \in \mathcal{N}_i$ receives the updated $\bar{x}_i$, updates the gradients, and sends them to its neighbors $k \in \mathcal{N}_j$. The variables and gradients for the rest of the nodes in the network are not changed by the update of node $i$.

*Coordinate descent equivalence and convergence.* Finally, we simply notice that, thanks to the consistency argument just shown, steps (4)-(5) correspond to steps (7)-(8). Thus, we have shown that our distributed algorithm implements the centralized coordinate method and therefore inherits its convergence properties. By invoking Theorem 4.3, the proof follows.

## V. Numerical simulations on a non-convex constrained quadratic program

In this section we present a numerical example showing the effectiveness of the proposed algorithm.

We consider an undirected connected Erdős-Rényi random graph $\mathcal{G}$, with parameter 0.2, connecting $N = 50$ nodes and we test the distributed algorithm on a partitioned non-convex constrained quadratic program in the form

$$\min_{x \in \mathbb{R}^n} \sum_{i=1}^{N} x_{\mathcal{N}_i}^\top H_i x_{\mathcal{N}_i} + r_i^\top x_{\mathcal{N}_i} + I_{X_i}(x_i), \quad (11)$$

where each $x_i \in \mathbb{R}$ for all $i \in \{1, \ldots, N\}$ and each cost matrix $H_i \in \mathbb{R}^{|\mathcal{N}_i| \times |\mathcal{N}_i|}$ is only symmetric (*not* positive definite). We construct $H_i$ as the difference between a positive definite matrix $\tilde{H}_i \in \mathbb{R}^{|\mathcal{N}_i| \times |\mathcal{N}_i|}$ and a suitable scaled version of the identity matrix. Finally, each function $I_{X_i}$ denotes the indicator function of the segment $X_i = [-\ell_i, u_i]$, i.e., we constrain each $x_i$ to lie into an interval. We set $\ell_i = -30$ and $u_i = 20$ for all $i \in \{1, \ldots, N\}$.

Problem (11) fits our set-up described in Section II by defining

$$f_i(x_{\mathcal{N}_i}) := x_{\mathcal{N}_i}^\top H_i x_{\mathcal{N}_i} + r_i^\top x_{\mathcal{N}_i}$$

and

$$g_i(x_i) := I_{X_i}(x_i) = \begin{cases} x_i & \text{if } \ell_i \leq x_i \leq u_i \\ +\infty & \text{otherwise.} \end{cases}$$

Moreover, we use the local approximation $q_i(s_i, \bar{x})$ as in (3) with $Q_i = \frac{1}{\alpha_i} I$ with $\alpha_i = 0.01$ for all $i \in \{1, \dots, N\}$.

In Figure 2 we plot the evolution of two selected components of the decision variable $x$ at each iteration $t$ (defined as discussed in Section IV), i.e., $x_i(t)$, $i = 14, 48$. The horizontal dotted lines represent the centralized solution. Since the algorithm is asynchronous and based on a coordinate approach, we plot the rate of convergence with respect to the normalized iterations $t/N$ in order to show the effective behavior with respect to the global time.

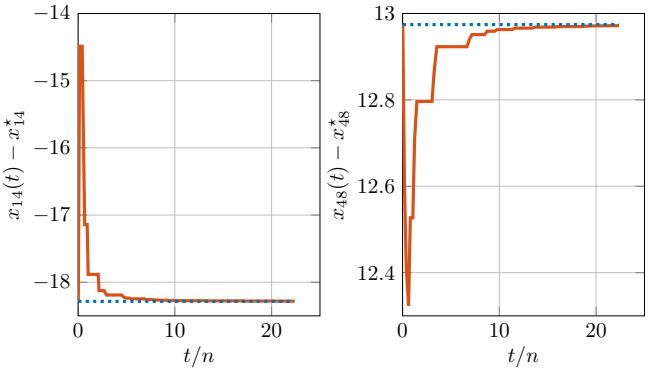


Fig. 2. Evolution of two decision variables $x_i$, $i = 14, 48$, for the distributed algorithm.

In Figure 3 we show the difference, in logarithmic scale, between the cost $V(x(t))$ at each iteration $t$ and the value of $V$ attained at the limit point $x^\star$ of $x(t)$ (proven to be a stationary point).

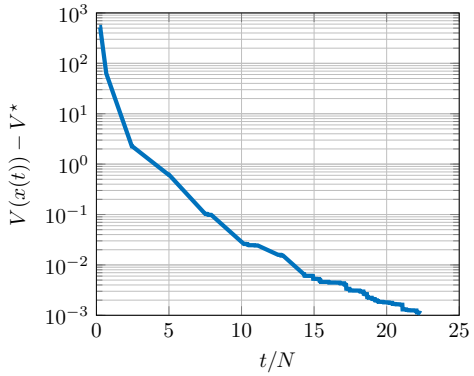


Fig. 3. Evolution of the cost error, in logarithmic scale, for the distributed algorithm.

## VI. Conclusions

In this paper we have proposed an asynchronous, distributed algorithm to solve partitioned, big-data non-convex optimization problems. The main idea is that each node updates its local variable by minimizing a suitable, local quadratic approximation of the cost, built via an information exchange with neighboring nodes. We prove the convergence of the distributed algorithm by showing that it corresponds to a proper instance of a coordinate descent method.


## Acknowledgments

The authors would like to thank Angelo Coluccia e Massimo Frittelli for their help and suggestions.


## References


[1] T. Erseghe, "A distributed and scalable processing method based upon admm," *IEEE Signal Processing Letters*, vol. 19, no. 9, pp. 563–566, 2012.

[2] R. Carli and G. Notarstefano, "Distributed partition-based optimization via dual decomposition," in *IEEE 52nd Annual Conference on Decision and Control (CDC)*, 2013, pp. 2979–2984.

[3] I. Necoara and D. Clipici, "Parallel random coordinate descent method for composite minimization: Convergence analysis and error bounds," *SIAM Journal on Optimization*, vol. 26, no. 1, pp. 197–226, 2016.

[4] M. Todescato, G. Cavraro, R. Carli, and L. Schenato, "A robust block-Jacobi algorithm for quadratic programming under lossy communications," in *IFAC-PapersOnLine*, vol. 48, no. 22. Elsevier, 2015, pp. 126–131.

[5] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Puschel, "Distributed optimization with local domains: Applications in MPC and network flows," *IEEE Transactions on Automatic Control*, vol. 60, no. 7, pp. 2004–2009, 2015.

[6] I. Notarnicola and G. Notarstefano, "Randomized dual proximal gradient for large-scale distributed optimization," in *IEEE 54th Conference on Decision and Control (CDC)*, 2015, pp. 712–717.

[7] F. Zanella, D. Varagnolo, A. Cenedese, G. Pillonetto, and L. Schenato, "Asynchronous Newton-Raphson consensus for distributed convex optimization," in *3rd IFAC Workshop on Distributed Estimation and Control in Networked Systems*, 2012.

[8] D. V. Dimarogonas, E. Frazzoli, and K. H. Johansson, "Distributed event-triggered control for multi-agent systems," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1291–1297, 2012.

[9] E. Wei and A. Ozdaglar, "On the $O(1/k)$ convergence of asynchronous distributed alternating direction method of multipliers," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, 2013, pp. 551–554.

[10] S. Lee and A. Nedić, "Asynchronous gossip-based random projection algorithms over networks," *arXiv preprint arXiv:1304.1757*, 2013.

[11] Y. Nesterov, "Efficiency of coordinate descent methods on huge-scale optimization problems," *SIAM Journal on Optimization*, vol. 22, no. 2, pp. 341–362, 2012.

[12] P. Richtárik and M. Takáč, "Parallel coordinate descent methods for big data optimization," *Mathematical Programming*, pp. 1–52, 2012.

[13] ——, "Iteration complexity of randomized block-coordinate descent methods for minimizing a composite function," *Mathematical Programming*, vol. 144, no. 1-2, pp. 1–38, 2014.

[14] I. Necoara, "Random coordinate descent algorithms for multi-agent convex optimization over networks," *IEEE Transactions on Automatic Control*, vol. 58, no. 8, pp. 2001–2012, 2013.

[15] P. Bianchi, W. Hachem, and F. Iutzeler, "A stochastic primal-dual algorithm for distributed asynchronous composite optimization," in *GlobalSIP*, 2014, pp. 732–736.

[16] A. Patrascu and I. Necoara, "Efficient random coordinate descent algorithms for large-scale structured nonconvex optimization," *Journal of Global Optimization*, vol. 61, no. 1, pp. 19–46, 2015.

[17] F. Facchinei, G. Scutari, and S. Sagratella, "Parallel selective algorithms for nonconvex big data optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 7, pp. 1874–1889, 2015.

[18] P. Di Lorenzo and G. Scutari, "NEXT: In-network nonconvex optimization," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 2, pp. 120–136, 2016.

[19] G. Binetti, A. Davoudi, D. Naso, B. Turchiano, and F. L. Lewis, "A distributed auction-based algorithm for the nonconvex economic dispatch problem," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 2, pp. 1124–1132, 2014.

[20] Y. Nesterov, "Gradient methods for minimizing composite functions," *Mathematical Programming*, vol. 140, no. 1, pp. 125–161, 2013.

[21] D. P. Bertsekas, *Nonlinear programming*. Athena scientific, 1999.